Technical Report 1351
July 1990

Engineering and Project Management Oriented Development System (EPOS)

Review and Analysis

R. Liu

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

The study covered in this document was conducted from August 1988 to October 1989. It was block funded out of Code 423 of the Naval Ocean Systems Center (NOSC), San Diego, California. The project no. is ECB3 03, agency accession no. DN088690, and program element 602234N. The work was performed by Code 411 of NOSC, San Diego, California.

Released by
D. L. Hayward, Head
Computer Systems Software
and Technology Branch

Under authority of
A. G. Justice, Head
Information Processing
and Displaying Division

RT

SUMMARY

OBJECTIVES

The objectives of this CASE experiment were to

- assist a Navy project in using a Computer-Aided Software Engineering (CASE) tool
- evaluate the CASE tool to determine if it provides the level of functionality the Navy application requires
- inform the vendor of what is lacking in the technology and thereby improve the tool's effectiveness for Navy use
- produce this report that describes the CASE experiment and documents its results.

RESULTS

A set of evaluation criteria was developed along with a rating scheme. The CASE tool was used experimentally in a project-specific context, and the results were evaluated according to the criteria. The tool was found to provide rigorous software methodologies to reinforce good software design practices, such as abstraction, decomposition, structuring, and information hiding. The software development process and configuration control are more rigorous by using the Engineering and Project Management Oriented Development System (EPOS) project and configuration management support facility. The project staff was pleased with the system design quality that resulted from using the tool and its underlying methodologies. However, EPOS lacks the user friendliness and robustness to make it wholly satisfactory. EPOS does not employ the power of modern workstations to provide a user interface that is (1) graphically oriented, (2) iconically animated, (3) has windowing, and (4) is pop-up/pull-down/pull-right menu driven.

RECOMMENDATIONS

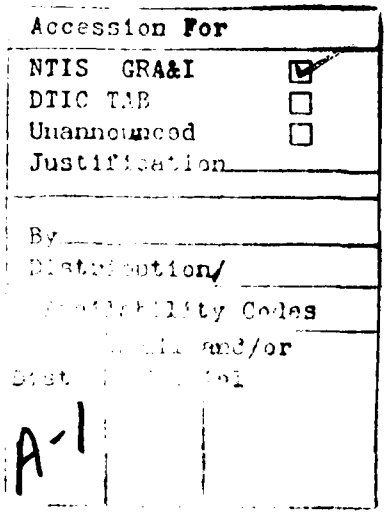
Based upon the results of this experiment, the following is recommended for the further development, use, and implementation of EPOS:

- use EPOS's rigorous software engineering notational schemes to serve as a common basis for communicating complex design information to designers and users. Through its explicit design rule checking, EPOS can reinforce modern software practices, such as abstraction, decomposition, structuring, and information hiding.
- use the underlying methodologies in EPOS to provide criteria to judge the completeness and quality of the design specification. However, EPOS may include a methodology guidance facility (an intelligent help system) to further assist users to comply with methodologies during the design process.
- EPOS can be much more effective if it reinforces software engineering methodologies while simultaneously providing the interactive graphical user interface needed to prevent the use of EPOS from slowing down the creative process. Successfully using the CASE application depends heavily on the power of tools and workstations to provide interactive graphics, iconic

interfaces, windowing, and context-sensitive menus. However, EPOS does not have these capabilities.

NOTE

The opinions expressed in this report are those of the author and do not reflect an official government position.



CONTENTS

CONTENTS (continued)

FIGURES

TABLE

1.0 CASE EXPERIMENT PROJECT BACKGROUND

Today's Navy projects are seeking to use Computer-Aided Software Engineering (CASE) tools because of their potential for improving the quality and lowering the cost of Navy software systems. Emerging CASE tools must be evaluated in a Navy context to determine and ensure that the tools provide the level of functionality Navy applications require. When CASE technology is evaluated early in system development, both the Navy and the CASE vendor benefit. The Navy determines what is lacking in the technology and provides this information to the vendor to promptly correct the deficiency, thereby making the tools more relevant for Navy use.

The Theater Communications Branch, Code 851, had a requirement for CASE tools. The Computer Systems Software and Technology Branch, Code 411, already investigating CASE tools, assisted Code 851 by demonstrating that the tools were available for immediate application. Code 411 also wanted to support the tool's early transition to Navy use.

2.0 CASE EXPERIMENT PROJECT GOALS

Personnel in the Special Security Communications Center (SSCC) project were motivated to use a CASE tool because they considered that the quality and responsiveness to change were the most critical success factors for the new system. They also realized that the traditional systems development process was inadequate. Project personnel needed to improve their design and analysis skills and utilize the tools to support new techniques. Moreover, an automated life-cycle support document generator was needed. Project personnel also felt that the use of computer-generated graphics documents was important, because communications using graphics, rather than words, would be more effective during the analysis and design phases of the project. (Note: The SSCC project was a project in Code 851 that was discontinued in 1989.)

3.0 SPECIAL SECURITY COMMUNICATIONS CENTER (SSCC) PROJECT BACKGROUND

The AN/MSC-63A Special Security Communications Center (SSCC) provided semiautomatic data communications support for processing record-sensitive, compartmented information (SCI) to United States Marine Corps (USMC) personnel operating in a tactical environment. The SSCC's hardware system consisted of USMC mobile shelters containing modems, cryptographic devices, displays, two digital computers (dual AN/UYK-44s), and data storage/retrieval devices. The SSCC's software system consisted of two components. The first component was a CMS-2M SDEX/M runtime executive to support realtime, multitask processing. The second component was the event-driven operational software, which had to meet stringent time constraints for performing realtime processing.

4.0 SELECTING A TOOL

The SSCC project staff soon realized that a project's use of software engineering disciplines and methodologies is far more important than using any specific CASE tool. They also agreed that tools, though important, are only useful when they support, simplify, and facilitate the application of disciplined approaches. Therefore, the first decision of the project staff was to use good software engineering disciplines coupled with the use of tools. The second decision was to identify specific project needs and to evaluate available tools to determine if a match could be found. Third, a potential match had to be analyzed in terms of risk, in order to confine selections within the constraints of project risk. The potential benefits envisioned from an ideal CASE toolset environment for the SSCC project included

- Support of standardized procedures and management policies to aid the development process.
- Enforcement of good software engineering practices.
- Improvement of productivity and software reliability through automation and process standardization and control.
- Improvement of software portability.
- Support of reuse of software design and code.
- Support of rapid prototyping/incremental development.
- Support of a standardized communication process to provide ongoing development information.
- The ability to coordinate, control, and automate the documentation process using SSCC's documentation standard, DoD-STD-2167A.

CASE toolsets are considered an emerging technology. As such, use of a CASE tool has certain associated risk elements. These elements of risk must be balanced with potential benefits. The perceived elements of risk for a CASE toolset used on the SSCC project were as follows:

- CASE tools are characterized as high cost items and can be computer-resource intensive. CASE tools often require graphical tool resources (graphics workstations), with some tools requiring graphics support throughout the development cycle. While this in itself is not a risk, it implies a procurement cycle in a government context that may cause schedule delays in both software and hardware deliveries.
- The decision to use improved software engineering disciplines created new problems and risks that had to be addressed and minimized. The existing knowledge of the system needed to be transitioned into the new tool, and personnel needed a "phase in" period to learn how to utilize the tool.
- A high-level design document of the SSCC project was not available. Therefore, "reverse engineering" had to be performed manually, using low-level documentation (source only).

The Engineering and Project Management Oriented Development System (EPOS) was initially identified as one of the more suitable CASE toolsets for experimental use at the time of the project (August 1988) for the following reasons:

- EPOS supports the integrated use of multiple software engineering methods, besides the Yourdon-Demarco approach, that include realtime methods, based on Petri nets and event/interrupts; as well as Parnas's module-oriented techniques, including module/functional breakdown and device-oriented methods.
- EPOS provides both multiple notations and the ability to use different methodologies during a session for a single system development. These notations include Jackson data-structure diagrams for data-driven modeling, hierarchy diagrams for top-down-driven modeling, data-flow diagrams for data-flow-driven modeling, flowcharts for function-driven modeling, Petri nets for event-driven modeling, and block diagrams for module-driven modeling. EPOS also provides bar/milestone charts, network diagrams, progress charts, responsibility/assignment matrices, and work breakdown structures for project management.
- EPOS integrates development and project management support based on interrelated modeling of project management and technical information. EPOS also provides computer-assisted project control and assessment.

- EPOS generates source code in all of the following standard languages: Ada, Pascal, C, and FORTRAN, and provides a code feedback mechanism to automatically update the design to agree with the code changes. EPOS runs on a variety of host computers currently used by NOSC and its contractors, including the Z-248 (IBM PC-AT compatible), Sun, Apollo, and DEC VAX.

5.0 APPROACH

The approach used for this effort involved three steps: (1) identify the key capabilities a CASE tool requires to support SSCC, (2) derive evaluation criteria to judge these capabilities, and (3) evaluate EPOS as a CASE tool within the context of an existing Navy application development project.

5.1 STEP 1—IDENTIFY KEY CAPABILITIES

Since SSCC employed a realtime multiprocessing system, the CASE tool had to provide the key capabilities to support the design, analysis, development, and documentation processes for environments involving distributed- and parallel-processing software development. The difficult, realtime system-design issues (such as representations for system behavior, critical timing, multitasking, and multiprocessor architecture) required support by the CASE design mechanism. Furthermore, successful use of the CASE tool had to be assessed using the following evaluation criteria:

- functionality
- power of tool
- ease of use
- robustness
- user support

5.2 STEP 2—DERIVED EVALUATION CRITERIA

The tool functionality criterion addresses multiple attributes. Since our experiment focused on the use of EPOS in the software design phase, we identified high-level coverage requirements, including support for

- formal design specification and validation.
- structure and module specification.
- data-dictionary specification and validation.
- interface specification and checking.
- programming design language (PDL) generation/validation.
- multiple-design methodologies.
- the DoD-STD-2167A documentation standard.

During our experiment, the power of each tool was assessed according to cost and schedule reductions by (1) how well the tool performed a specific engineering task, (2) how quickly it did so, (3) how easily it supported multiple users, and (4) how well it shared information across users and across tasks.

Factors affecting a tool's ease of use were (1) intuitiveness, (2) tailorability, (3) intelligence and helpfulness, (4) predictability, and (5) error handling.

The robustness of a CASE tool consists of (1) trouble report history, (2) consistency of the tool, (3) fault tolerance, and (4) self-instrumentation.

Finally, the quality of user support was assessed using the following criteria: (1) tool and vendor history; (2) purchasing, licensing, and maintenance; (3) support personnel; (4) user-group feedback; (5) installation and system integration; (6) training; and (7) documentation.

5.3 STEP 3—EVALUATE EPOS AS A CASE TOOL

This step involved evaluating EPOS as a CASE tool within the context of an existing Navy application development project. It entailed actually applying EPOS to the SSCC software development process.

5.4 OVERVIEW OF EPOS

EPOS is a software development tool that includes the following four components:

1. EPOS-S—a design specification language that specifies both high- and low-level design.
2. EPOS-T—a document specification language that generates software design documentation.
3. EPOS-P—a specification language that automates project management and configuration control.
4. EPOS RE-SPEC—a reverse-engineering tool that recreates the EPOS design specification (EPOS-S) from a programming language source code.

6.0 FINDINGS

6.1 USE OF THE EPOS DESIGN SPECIFICATION LANGUAGE (EPOS-S)

6.1.1 General Description

EPOS-S was used to create a high-level SSCC system design and detailed design for selected system components, such as an asynchronous driver and a synchronous driver. Results showed that EPOS-S was suitable for all levels of the design (both high level and low level) and that it was useful for designing components such as device drivers and realtime processing algorithms. Moreover, EPOS's functionality was versatile enough to handle AN/MSC-63A (SSCC).

The following example illustrates how to use EPOS-S to specify partitioning of hardware/software functions and realtime system behaviors, such as critical timing, multitasking, and multiprocessing. The SSCC's operational Message Processing (MSGPRO) program performed such functions as link coordination, message protocol control, message validation, message reformatting, message distribution, and so on. These functions were independent processing entities that executed simultaneously on two AN/UYK-44 processors. Using the EPOS-S design object "ACTION" (figure 1), software partitioning was accomplished by designating concurrent tasks. Hardware partitioning was accomplished by assigning concurrent tasks onto desired CPUs and defining the parallel processing of the system. The designer could also designate synchronization and communication mechanisms between objects (mutual exclusion and rendezvous), as well as prioritized concurrent tasks. Finally, the EPOS document generator could display

these partitioning assignments and system behaviors in a Petri net diagram (figure 2). The design specification was also supported by automated design rule checking for analyzing synchronization and determining the presence of deadlocks, completeness, type conflicts, etc.

```
ACTION PROCEDURE MSGPRO . (Mssage Processing Program)
DESCRIPTION :
PURPOSE :
"The MSGPRO program performs the operational message processing functions of the SSCC. These functions include Link Coordi-
nation, Message Protocol Control, Message Validation, Message Archive, Report Generation, Message
Retrieval, Configuration Control, etc.".
CATEGORY : 'MSGPRO' .
DESCRIPTIONEND
DECOMPOSITION :
 PARALLEL    (   AD , AR , AT , CC , CRCKPW , CRCNGPW , CRCNVRT , CRDIST ,
                 CREDIT , CRENCRP , CRGENPW , CRSTRIP , CRSVCGEN , DH , DK ,
                 IC , IH , LM , MA , MD , ME , MM , MP , MQ , MR , PH , PI ,
                 PM , PO , RG , RI , RO , RR , SD , SR , ST , TC , TH , TM ,
                 VH , VI , VO ) .
SYNCHRO : EXCLUSIV ( AD REENTRANT PRIO , SD ENENTRANT ) .
PERFORMANCE :
                 " FOR AR (Asynchronous Receive) I/O Response Block Task,
                 execution time to form complete messages from an incoming
                 message block is less than 1 second.
                 For AD (Asynchronous Driver) use 5000 " .
PROCESSED :
                 " First AN/UYK-44 processor " .
ACTIONEND
```

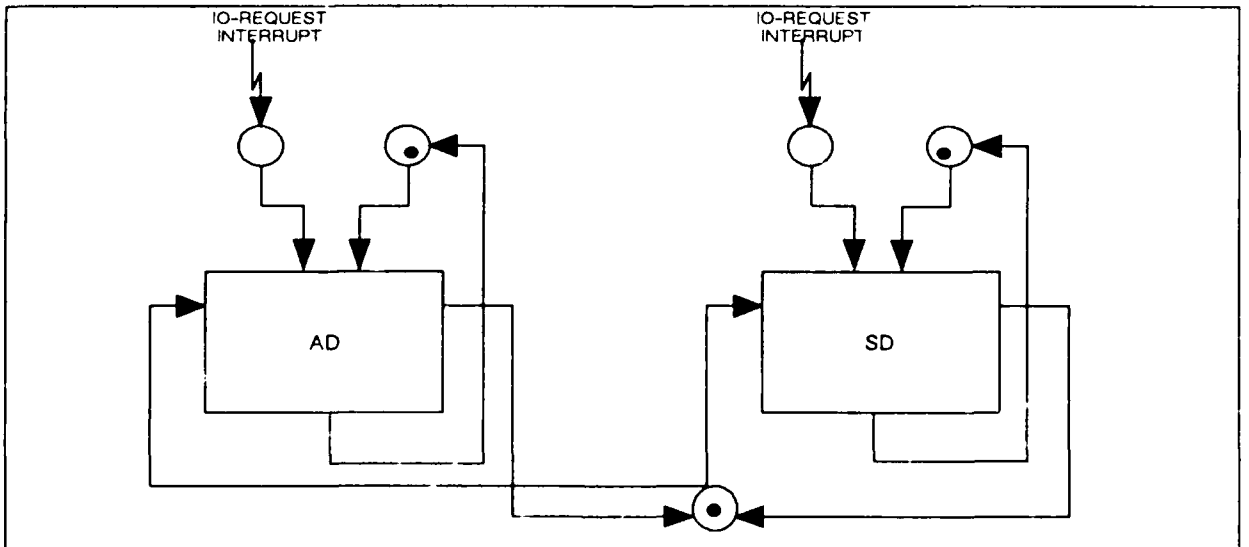
The PARALLEL keyword indicates the subactions "AD", "AR", "AT", "AC", etc., are preformed concurrently, with respect to the control flow.

The SYNCHRO keyword indicates AD(Asynchronous Driver) and SD (Synchronous Driver) cannot take place simultaneously. If both tasks are triggered at the same time, then AD task has priority over the SD task.

The PERFORMANCE keyword specifies performance requirements for AR's execution time and AD task's memory allocation (it allocates 5000 memory units for AD).

The PROCESSED keyword specifies our two-processor configuration (dual AN/UYK-44 configuration). The MSGPRO program is to be performed in the first AN/UYK-44 processor.

Figure 1. EPOS design object "ACTION."



A Petri net generated by the EPOS documentation tools, based upon the EPOS-S description.

Figure 2. EPOS Petri net diagram.

6.1.2 PDL Generation from EPOS-S Specifications

After EPOS-S had created tasks, data, and procedural abstractions, we used the EPOS code generator to generate Ada PDL from our EPOS-S specification. The automated PDL provided legal Ada programming construction as specified in the language reference manuals (DOD83). We also used the Ada compiler to perform type checking of all interfaces between packages, subprograms, and tasks, reducing time spent on debugging. This automated Ada PDL was excellent for design reviews and reports. Figure 3 is an example of an EPOS-generated Ada PDL.

6.1.3 Results of Using EPOS-S

Project personnel felt that by using EPOS-S's rigorous notational schemes, we acquired a common basis for communicating complex design information between designers and users. Furthermore, EPOS-S's explicit design-rule checking reinforced good design practices, such as information hiding, process abstraction, decomposition, and module structuring. We also found that the graphics-oriented approach during the design phase was essential for providing effective communication between analysts, designers, programmers, and users. The EPOS graphics generator aided us greatly in this experiment. Perhaps more important, the design modifications requested due to the changing system requirement needed much less effort to complete than during our previous experience using other development techniques.

In our opinion, the use of EPOS-S and its underlying methodologies did not constrain our creativity during the design phase. On the contrary, we were relieved of many of the more tedious tasks, such as producing hand-drawn models and writing extensive descriptions. Consequently, we became more productive and could concentrate on the more important design issues, such as system partitioning and critical timing.

6.2 USE OF EPOS DOCUMENT GENERATOR (EPOS-T) TO GENERATE DoD-STD-2167A SOFTWARE DESIGN DOCUMENT

6.2.1 General Description of Use

An example of the DoD-STD-2167A Software Design Document (DID No. DI-MCCR-80012A) for SSCC was generated using the EPOS document specification language EPOS-T, which is a search-and-query language. It provides options to search the EPOS project database and sort by database entries to define a series of documentation segments—and include general descriptions and diagram data. Since all interrelated representations (requirements, system design, detailed design, etc.) were available in the EPOS database, the specific documentation standards, such as DoD-STD-2167A, could be generated from the current database and not from possibly outdated diagrams and files. Our first step was to create the overall documentation specification for the Software Design Document (SDD). Then the cover sheets, footers, headers, and the appropriate outline (e.g., of the Data Item Descriptions in DoD-STD-2167A) were also specified using EPOS-T. Figure 4 shows an example of specifying the SDD document, and figure 5 presents an excerpt of the SDD Detailed Design section for the SSCC project generated by the EPOS document generator.

6.2.2 Results of Using EPOS-T

In conclusion, we found the EPOS document generator (EPOS-T) had the flexibility to generate project documentation in a project-independent fashion. However, we hope that templates of DoD-STD-2167A will be provided in the future to assist users in developing documentation according to Data Item Descriptions.

```
-- This is Asynchronous Driver Subprogram (AD) which shall perform the AN/UYK-44 machine level
-- input/output processing for the asynchronous mode external channel interfaces. For further details
-- and descriptions, see AD Functional Specification.

generic
type I/O_Block_Input is private ; -- I/O block used as input to AD
                                     (Asynch Drive)
type I/O_Block_Output is private ; -- I/O block used as output from AD

package Asynchronous_Driver is -- package specification starts here

procedure I/O_Request ( Data_in : in I/O_Block_Input ;
                        Data_out : out I/O_Block_Output ) ;
-- The purpose of this procedure is to perform I/O request processing. It receives requests from a
-- specific channel, processes them, and then returns control to the subprogram called SDEX/M
-- based upon the occurrence of I/O interrupts and/or timeouts.

private

type I/O_Block_Input is -- I/O block record for input
    record
        Function Code : TED ; -- type of I/O request
        Physical Channel : TBD ; -- physical channel for I/O
                                    request
        Reinitiate_Input_Flag TBD ; -- flag to control input
        Buffer_Address : TBD ; -- output buffer address
        Buffer_Length : TBD ; -- output buffer in bytes
        Baud_Rate : TBD ; -- data transmission rate in bps
        Character Coding : TBD ; -- type of message characters
        Parity : TBD ; -- parity setting for the channel
        Stop_Bit : TBD ; -- stop bits for the channel
end record ; --

type I/O_Block_Output is -- I/O block record for output
    record
        Caller_Task_ID : TBD ; -- ID number of the caller
        I/O_Complete_Status : TBD ; -- status indicating result
                                       of request
        Buffer_Address : TBD ; -- buffer address
        Buffer_Length : TBD ; -- buffer length in bytes
end record ;
end Asynchronous_Driver ; -- end of Asynchronous_Driver
package body Asynchronous_Driver is

task IOC_Discrete_Interrupt  -- These interrupts and timeouts
processings
    end IOC_Discrete_Interrupt;

task Crypto_Delay_Timeout is -- has the duty of controlling the
I/O request processing.
     end Crypto_Delay_Timeout;

task IOC_Input_Chain_Interrupt is
    entry Input_Chain_Interrupt;
    for Input_Chain_Interrupt use at Input_Chain_Interrupt_Address;
    -- Calls
       -- Entry call to consumer process
end IOC_Input_Chain_Interrupt ;
```

Figure 3. EPOS-generated Ada PDL.

```
task Input_Buffer_Timeout
    end Input_Buffer_Timeout;

task IOC_Output_Chain_Interrupt
    entry IOC_Output_Chain_Interrupt;
    for IOC_Output_Chain_Interrupt use at Output_Chain_Interrupt_Address;
    -- Calls
       -- Entry call to consumer process
end IOC_Output_Chain_Interrupt;

task Output_Buffer_Timeout
     end Output_Buffer_Timeout;

task body IOC_Discrete_Interrupt is separate;
task body Crypto_Delay_Timeout is separate ;
task body IOC_Input_Chain_Interrupt is separate;
task body Input_Buffer_Timeout is separate;
task body IOC_Output_Chain_Interrupt is separate;
task body Output_Buffer_Timeout is separate;

-- This package is designed for a real-time application that the
-- size of the program is very critical. Therefore, it is designed
-- without using the message buffering technique.
end Asynchronous_Driver;
```

Figure 3. EPOS-generated Ada PDL (continued).

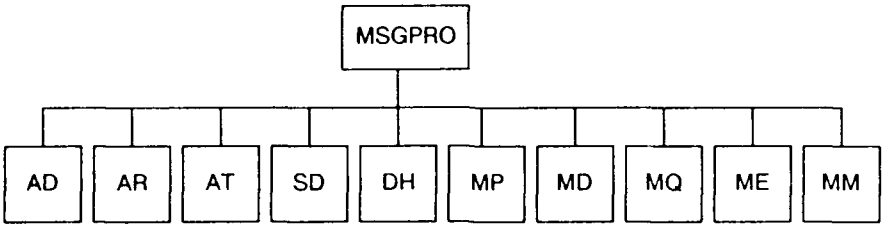
```
SECTION(4; "Detailed Design";)
SECTION(4.1; "SSCC OP S/W – MSGPRO";)
WRITE"The MSGPRO program shall perform the operational message processing functions of the SSCC.
These functions include Link Coordination, Message Protocol Control, Message Validation, Log, Message
Archive, Report Generation, Message Retrieval, Configuration Control, Training Simulation and the back-
ground and operational test functions of System Test.":)

USING ALL 'ACTION' WITH 'MODULE' IN DESIGN DO
##
##    means :  select all design objects that are actions
##             with sub type MODULE
##
##


     [ FOR DESIGN-LEVEL IN 1 TO MAXLEVEL DO
         USING ALL ON LEVEL DESIGN-LEVEL IN USE DO

##   means :  search all design levels for the above
##            mentioned class of design objects

     [ SORT-BY SORT-KEY

##   means :  bring them into alphabetical order as per
##            name for output

     FOR OBJ IN 1 TO MAXOBJECT DO
      [
       NAME : = GETVALUE ( OBJ, 'NAME' )
       SECTION (4.1, OBJ; NAME; )
       EXEC-TEXT PURPOSE-ONLY (NAME, $)
         COLUMN TEXTCOLUMN

##   means :  determine name of the next object of selected
##            class, output a headline for the object stating
##            its name execute the macro purpose-only with
##            its two parameters
```

Figure 4. Example of a 2176A SDD document specification.

SSCC DoD-STD-2167A(SDD)
8-8-87

4. Detailed Design

The MSGPRO program shall perform the operational message processing functions of the SSCC. These functions include Link Coordination, Message Protocol Control, Message Validation, Log, Message Archive, Report Generation, Message Retrieval, Configuration Control, Training Simulation and the background and operational test functions of System Test.



Operational Message Processing Program (MSGPRO) Hierarchy

4.1 Asynchronous Driver (AD)

AD performs the machine level input and output processing for the asynchronous mode external channel interfaces. AD ensures that the input data is associated with the correct channel and passes the input data to the Asynchronous Receive (AR) subprogram. AD also multiplexes the output request to the correct interface.

4.2 Asyncrhonous Receive (AR)

AR accepts input data from AD. Except for the COMP message, the input data shall be passed to the Message Examiner (ME) subprogram.

Figure 5. Excerpt of the 2167A SDD detailed design section for the SSCC project generated by the EPOS document generator.

6.3 USE OF EPOS SPECIFICATION LANGUAGE EPOS-P TO AUTOMATE PROJECT MANAGEMENT AND CONFIGURATION CONTROL

6.3.1 General Description and Use

The specification language EPOS-P is used to describe information related to project management and product configuration control. Its basic language elements are so-called "management-objects"—the equivalents to the design objects in EPOS-S. Figure 6 gives an overview of the available types of management objects.

A faculty of every configuration management system is the capability to report and track errors, as well as to propose, evaluate, and include approved parts of the product. From a configuration management point of view, these parts are baselines that are established at certain times, e.g., at the end of a project phase after a review. They consist of a certain agreed upon version of the development results. Any subsequent changes of these baselines—starting with a proposal for a change and ending in its implementation and test—must be carefully controlled.

Management Object Type	Information Described
ACTIVITY	Work Breakdown Structure PERT Planning
TEAM MEMBER	Project Organization Structure Functions and Responsibilities
PROGRESS-REPORT	Report on Progress, Trouble Status Changes
CHANGE-PROPOSAL	Change Proposal and Evaluations
ERROR-REPORT	Reports on Errors

Figure 6. Overview of available types of management objects.

EPOS-P provides management objects of types ERROR-REPORT and CHANGE-PROPOSAL to specify any error and to formulate, justify, and evaluate suggested modifications. Figure 7 shows an example of a proposed change in a project.

```
CHANGE-PROPOSAL EXTENSION-ERROR-TRACING.

CATEGORY: 'process start', 'critical', 'general'.

IDENTIFICATION: 'Xenus Ltd. (Peter Joans) 2/86'

CHANGE:  "Not only static sources of errors, but also dynamic ones should be
          considered, especially during close-down."

REFERENCE:  ERROR-5-85,
            ERROR-6-85

CONCERNS: CONSIDERS P5.2, P5.3

REASON:  "Security can not be assured if data will be lost during close-down."

CHANGE-PROPOSALEND
```

Figure 7. EPOS-P change proposal.

All objects in EPOS can exist in different versions—as different changes in time and/or variants as different forms at the same time.

The granularity of the version/variant control extends from single unstructured information objects, as the smallest ("atoms"), to parts of a representation or representations, consisting, themselves, of a structure of specification objects (e.g., a module). Therefore, at one end of the scale, the developers are assured by keeping track of their (atomic) changes; whereas, at the other, configuration management can

trace versions/variants of configuration items that are normally structured objects. Figure 8 shows documentation of existing versions of single unstructured objects.

Design Object	Version	Author	Date
INPUT-CONTROL-VARIABLE	2.2.0 2.1.2 2.1.0	Liu Sutton Tran	12-17-89 12-18-89 12-20-89
READ-MEASURED-VALUE	2.2.4 1.1.3 1.1.2	Smith Jackson Liu	11-02-89 12-03-89 12-02-89

Figure 8. Documentation of existing versions.

Besides the pure administration of versions/variants of configuration items, there are also the procedural aspects of identifying baselines and formal control of changes to these baselines.

A baseline is an (intermediate) project result, "frozen" at some point in time to form a fixed-reference configuration. Of key importance are the different representations of the system (requirement specification, design, source code, etc.). These, or at least part of these, will be identified as baselines. During development, and especially during maintenance, requests for changes can occur on different levels of representations—with differing significance. If new requirements or errors in a representation require a change in a representation that is already baselined, EPOS enforces a configuration management procedure. EPOS baseline status accounting includes the automatic documentation of all baselines, and all change proposals and their current status, as well as details on the responsible project team members. After completion of the change activity, a new baseline can be fixed.

6.3.2 Results of Using EPOS-P

The experiment team felt that during the software design phase, EPOS had aided the project significantly. This section details what occurred during the software design phase.

The computer environment used for this experiment was typical of many such installations—a combination of a local-area networked Sun Workstation connected to a VAX mainframe. The main problem our design team had before using a CASE environment was coordinating and controlling design data across the workstations and among designers. Typical difficulties the design group encountered during the design phase were

- A new version of a design library component was released halfway through the project, but all the designs that used this component were not updated.
- A software engineer modified a component "on the fly" to complete a target simulation on the mainframe. The changes were not reflected in the design document. This oversight resulted in inconsistencies in the design; i.e., different versions of the same component were used.
- Product release took more time than expected because of time spent tracking down the correct versions and resolving the associated integration problems.

Our solution was to use the EPOS-P project specification language to control the version and variant administration of the design documents. By using EPOS-P and EPOS-M, software design releases were controlled through EPOS. Whenever a designer was ready to release a subsystem, a new configuration was created by activating EPOS-P's CHANGE-PROPOSAL attribute. Only the latest released configuration could be used for sharing intermediate designs with other designers. The project database always contained the most recent versions of all designs. The system bound all the current versions of the design files defined by EPOS-P into a new version of the database. This procedure eliminated the frustrations and possible errors of manually looking for the current versions of various files.

6.4 USE OF EPOS RE-SPEC TO REVERSE ENGINEER THE DESIGN OF SSCC

6.4.1 General Description and Use

EPOS RE-SPEC is a reverse-engineering tool for re-creating the EPOS design specification (EPOS-S) from a programming language source code.

Since a substantial body of design and software had already been written for SSCC before EPOS was used, the ability to perform reverse engineering was a critical feature. EPOS provides a reverse-engineering tool (RE-SPEC) to re-create EPOS design specifications from a programming language source code.

Using EPOS RE-SPEC to reestablish the SSCC design specifications consisted of the following activities:

a. Formal source code analysis

b. Functionality determination

c. Establishment of interrelated specifications

d. Verification

The **formal source code analysis** was used to capture all basic software structures, such as the hierarchical calling structure, module interfaces, internal/external data structure, control flow and data flow, interfaces to language-dependent library routines, and low-level input/output functions. The **functionality determination** was used to capture software structures from another spectrum of information sources, such as low-level design representations and internal documentation (e.g., comments or names/labels) within the source code. The process of **establishment of interrelated specifications** was used to form links between the different descriptions available as project documentation. It also updated the specifications so they reflected the current status of the software represented by the source code. The process of **verification** of the results of the reverse engineering was essential. After assuring completeness and consistency of the representations, themselves, source code in the original programming language was regenerated from the EPOS-S design specification and compared for functional equivalence with the original piece of software. EPOS RE-SPEC automated all of the above described steps.

6.4.2 Results of Using EPOS RE-SPEC

Unfortunately, we were unable to have any hands-on experience using RE-SPEC during our experiment, because, at that time, RE-SPEC did not support CMS-2.

6.5 EPOS' PROBLEMS ENCOUNTERED

6.5.1 User Interface Problems

EPOS has several user interface problems that can result in frustration, delays in generating diagrams, and lost work. These interface problems are described in the following paragraphs.

6.5.1.1 **EPOS Response Speed.** EPOS has a slow response time. Under the Sun3 workstation configuration, when the "menu input" is entered, several seconds may pass before a new menu appears. This delay may cause the user to conclude that the system has crashed. The user may then try several input/enter commands to revive the system. Then, when a menu finally responds, processing that menu may cause the loss of a design-object input or inactivate a terminal. The slow response time is not only frustrating, but contributes to delays in creating documents.

6.5.1.2 **User Data Entry.** The data entry method for EPOS is difficult to use. EPOS' menu interface sometimes frustrates the user because it focuses field paths or presents inconsistent information (see figure 9).

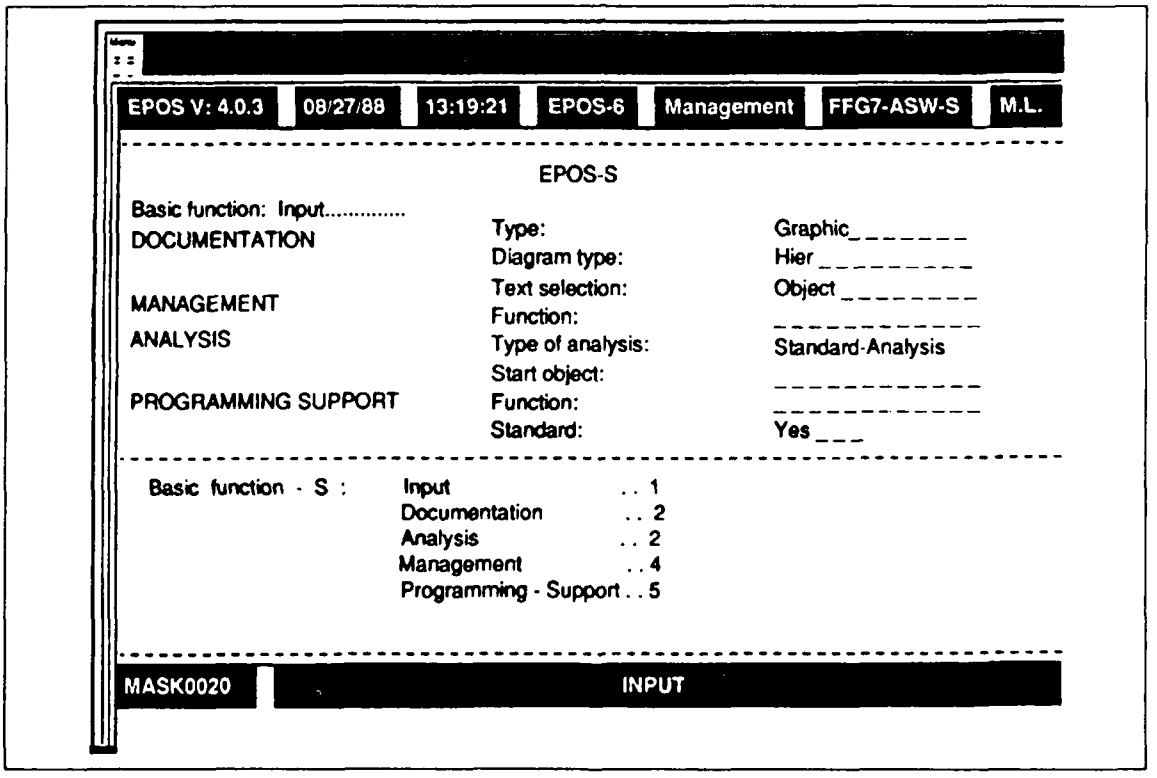


Figure 9. Example of an EPOS menu interface.

6.5.1.3 **Error Recovery.** EPOS does not offer the user a means of graceful recovery from unexpected errors, and error messages are unintelligible to the average user (see figure 10).

Figure 10. Example of an EPOS unintelligible error message.

6.5.1.4 **Graphics Interface.** In order to specify design graphics, we have to purchase third-party software. Once EPOS-S generates the graphics, we cannot modify the existing graphics output, because the graphics editor is not completely integrated with EPOS-S.

6.6 USER SUPPORT

Vendor support in this country is inadequate. The more experienced support engineers are not in this country, and the support staffs that are here can answer only simple questions. EPOS should provide a higher level of user support.

7.0 EVALUATION RATING

Five key capabilities were evaluated as specified in paragraph 5.1. An evaluation rating scheme of 1 through 5 was developed, where 5 represents an outstanding capability, and a 1 represents a minimal or nonexistent capability. Table 1 lists the results of this evaluation rating.

8.0 CONCLUSIONS AND RECOMMENDATIONS

8.1 CONCLUSIONS

EPOS was sufficiently functional to handle the AN/MSC-63A (SSCC). We feel that the underlying methodologies in EPOS reinforced good software design practices and provided criteria to judge the

Table 1. EPOS evaluation rating.

CAPABILITIES	EPOS-S Design Specification					EPOS-P Project Management and Configuration				
SCALE	1	2	3	4	5	1	2	3	4	5
A. Functionality										
1. Design specification and validation					X					N/A
2. Structured module specification					X					X
3. Data dictionary specification and validation					X					X
4. Interface specification and checking					X					X
5. Programming design language				X						N/A
6. Multiple design methodologies					X					
7. 2167A documentation support				X				X		
B. Power of Tool										
1. Performance quality		X					X			
2. Speed		X					X			
3. Friendliness		X					X			
4. Multi-user support		X					X			
5. Information sharing		X					X			
C. Ease of Use										
1. Intuitiveness		X					X			
2. Tailorability		X					X			

Table 1. EPOS evaluation rating (continued).

CAPABILITIES	EPOS-S Design Specification					EPOS-P Project Management and Configuration				
SCALE	1	2	3	4	5	1	2	3	4	5
C. Ease of Use (continued)										
3. Intelligence of helpfulness		X					X			
4. Predictability			X					X		
5. Error handling		X					X			
D. Robustness										
1. Consistency		X					X			
2. Fault tolerance	X					X				
3. Self-instrumentation	X					X				
E. User Support										
1. Tool and vendor history		X					X			
2. Purchasing, licensing, maintenance			X					X		
3. Support personnel		X					X			
4. User-group feedback	X					X				
5. Installation		X					X			
6. Training		X					X			
7. Documentation		X					X			

completeness and quality of the system design specification. However, EPOS' lack of user friendliness and robustness made it not fully satisfactory. It does not utilize the power of modern workstations to provide a user interface that is graphically oriented, iconically animated, has windowing, and is pop-up/pull-down /menu driven. In the final analysis, by using EPOS, the SSCC project benefited in terms of system quality. We did not expect much productivity gain in the design phase. A thorough evaluation of the effectiveness of a CASE environment cannot be conducted until the maintenance phase. Furthermore, we feel the CASE environment is dynamic and requires ongoing monitoring and enhancement.

8.2 RECOMMENDATIONS

EPOS can be used to provide criteria to judge the completeness and quality of a design specification. In addition, the rigorous software engineering notational schemes provided by EPOS can serve as a common basis for communicating complex design information to designers and users. However, to make EPOS much more effective, it should provide interactive graphics, iconic interfaces, windowing, and context-sensitive menus. These features would enable EPOS to reinforce software engineering methodologies while simultaneously providing the interactive capabilities required to prevent its use from slowing down the creative process.

9.0 BIBLIOGRAPHY

Lempp, P. 1988. "Support Environment Concepts for Cost-Effective Transition to Ada Technology," *Proc. Sixth National Conference on Ada Technology*. 14–17 March 1988, Arlington, VA.

Software Products and Services, Inc. "EPOS Reference Manual," version 4.0.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1 AGENCY USE ONLY *(Leave blank)*	2 REPORT DATE	3 REPORT TYPE AND DATES COVERED
	July 1990	Final: FY88 – FY89

4 TITLE AND SUBTITLE
ENGINEERING AND PROJECT MANAGEMENT ORIENTED DEVELOPMENT SYSTEM (EPOS)
Review and Analysis

5 FUNDING NUMBERS
PE: 602234N
PR: ECB3 03
WU: DN088690

6 AUTHOR(S)
R. Liu

7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152–5000

8 PERFORMING ORGANIZATION REPORT NUMBER
NOSC TR 1351

9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152-5000

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

11 SUPPLEMENTARY NOTES

12a DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b DISTRIBUTION CODE

13 ABSTRACT *(Maximum 200 words)*

An experiment was conducted to assist a Navy project in using and evaluating a Computer-Aided Software Engineering (CASE) tool to determine if it provided the level of functionality the Navy's application required. Any technological deficiencies were to be reported to the vendor for correction to improve the tool's effectiveness for Navy use. The CASE tool selected for this experiment was the Engineering and Project Management Oriented Development System (EPOS) because of its multiplicity of uses within the context of an existing Navy application development project. A set of evaluation criteria was developed along with a rating scheme.

The tool was found to provide rigorous software methodologies to reinforce good software design practices, such as abstraction, decomposition, structuring, and information hiding. By using the EPOS project and configuration management support facility, the software development process and configuration control are more rigorous. The project staff was pleased with the quality of the system design that resulted from the tool and its underlying methodologies. However, EPOS's lack of user friendliness and robustness cause it to be not totally satisfactory. That is, EPOS does not utilize the power of modern workstations to provide a user interface with graphic orientation, iconical animation, windowing, and a pop-up/pull-down/pull-right menu.

14 SUBJECT TERMS
computer-aided software engineering (CASE) tool
engineering and project management oriented development system (EPOS)
software engineering methodologies

15 NUMBER OF PAGES
25

16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT